Spring Form Tag Library

One of the view technologies you can use with the Spring Framework is Java Server Pages (JSPs). The Spring Web MVC framework provides a set of tags in the form of a tag library, which is used to construct views (web pages). The Spring Web MVC integrates spring's form tag library. Spring's form tag accesses to the command object, and also it refers to the data our spring controller deals with. A **Command** object can be defined as a JavaBean that stores user input, usually entered through HTML form is called the **Command** object or **Model** Object. The spring form tag makes it easier to develop, maintain, and read JSPs. The spring form tags are used to construct user interface elements such as text and buttons. Spring form tag library has a set of tags such as **<form>** and **<input>**. Each form tag provides support for the set of attributes of its corresponding HTML tag counterpart, which allows a developer to develop UI components in JSP or HTML pages.

Configuration – spring-form.tld

The spring form tag library comes bundled in **spring-webmvc.jar**. The **springform.tld** is known as **Tag Library Descriptor** (**tld**) file, which is available in a web application and generates HTML tags. The spring form tag library must be defined at the top of the JSP page. The following directive needs to be added to the top of your JSP pages, in order to use spring form tags from this library:

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
```

Here, **form** is the tag name prefix, which will be used for the tags from this spring form tag library in JSP pages.

The following table shows few important tags of the spring form tag library:

Tag	Description
form:form	Generates the HTML **<form>** tag. It has the name attribute that specifies the command object that the inner tags should bind to.
form: input	Represents the HTML input text tag.
form:password	Represents the HTML input password tag.
form:radiobutton	Represents the HTML input radio button tag.
form: checkbox	Represents the HTML input checkbox tag.
form:select	Represents the HTML select list tag.
form:options	Represents the HTML options tag.
form:errors	Represents the HTML span tag. It also generates span tag from the error created as a result of validations.

The difference between Html tags and Spring form tags

Tags	HTML	Spring
Input	<input type = "text">	<form:input>
Radiobutton	<input type="radiobutton">	<form:radiobutton> <form:radiobuttons> Specify list of values for radiobuttons in one shot
Checkbox	<input type="checkbox">	<form:checkbox> <form:checkboxes> Specify list of values for checkboxes in one shot
Password	<input type="password">	<form:password>
Select and option	<select name="course"> <option value="java">java</option> <option value="spring">spring</option> </select>	<form:select name="course"> <form:option value="java" label="java"/> <form:option value="spring" label="spring"/> </form:select> <form:options> Specify the list of options in one shot
Text area	<input type="textarea">	<form:textarea>
Hidden	<input type="hidden">	<form:hidden>

The <form: form> tag

The **<form: form>** tag is used to generate an HTML **<form: form>** tag in any of the JSP pages of a Spring application. It is used for binding the inner tags, which means that all the other tags are the nested tags in the **<form:form>** tag. Using this **<form: form>** tag, the inner tags can access the **Command** object, which resides in the JSP's **PageContext** class.

```
<form:form method="POST" modelAttribute="user" action="register">
        ........
</form: form>
```

The commandName & modelAttribute attribute

The attributes **commandName** and **modelAttribute** on the form:form tag do primarily the same thing, which is to map the form's fields to an Object of some type in the Controller. I believe modelAttribute is the preferred method, and commandName is only there for backwards compatibility.

The <form: input>tag: The **<form: input>** tag is used for entering the text by the user in any of the JSP pages of the Spring web application. The following code snippet shows the use of the **<form: input>** tag in JSP pages:

```
<form:form method="POST" modelAttribute="user" action="register">
    <table>
      <tr>
        <td>Enter your name:</td>
        <td><form:input path="name" /></td>
      </tr>
      <tr>
        <td>Enter your mail:</td>
        <td><form:input path="email" /></td>
      </tr>
    </table>
</form:form>
```

The path attribute

The <form: input> tag renders an HTML <input type="text"/> element. The path attribute is the most important attribute of the input tag. This path attribute binds the input field to the form-backing object's property.

Let's take an example, if user is assigned as modelAttribute of the enclosing <form/> tag, then the path attribute of the input tag will be given as name or email. It should be noted that the User class contains getter and setter for name and email properties.

The <form:checkbox> tag

The **<form:checkbox>** tag is same as the HTML **<input>** tag, which is of the checkbox type.

```
<form:checkbox path="skills" value="Excel" label="Excel"/>
<form:checkbox path="skills" value="Word" label="Word"/>
<form:checkbox path="skills" value="Powerpoint"  label="Powerpoint"/>
```

In the preceding code snippet, the User class property called skills is used in the checkbox option. If the skills checkbox is checked, the User class's skills property is set accordingly.

The <form: radiobutton> tag

The **<form: radiobutton>** tag is used to represent the HTML **<input>** tag with the **radio** type in any of the JSP pages of a Spring web application. It is used when there are many tag instances having the same property with different values, and only one radio value can be selected at a time. The following code snippet shows how we use the **<form:radiobutton>** tag in JSP pages:

```
<tr>
  <td>Gender:</td>
  <td>
    Male: <form:radiobutton path="gender" value="male"  label="male"/>
    Female: <form:radiobutton path="gender" value="female"  label="female"/>
  </td>
</tr>
```

The <form : password> tag

The **<form: password>** tag is used to represent the HTML **<input>** tag of the **password** type, in any of the JSP pages of a Spring web application. By default, the browser does not show the value of the password field. We can show the value of the password field by setting the **showPassword** attribute to **true**. The following code snippet shows how we use the **<form: password>** tag in the JSP page:

```
<tr>
  <td>Password :</td>
  <td>
     <form:password path="password" />
  </td>
</tr>
```

The <form: select> tag

The **<form: select>** tag is used to represent an HTML **<select>** tag in any of the JSP pages of a Spring application. We use the **<form: select>** tag for binding the selected option with its value. The **<form: option>** tag is the nested tag in the **<form:select>** tag. The following code snippet shows how we use the **<select>** tag in the JSP pages:

```
<tr>
  <td>Department</td>
  <td>
     <form:select path="department" items="${departmentMap}"  />
  </td>
</tr>
```

Note: Here departmentMap is collection with multiple keys and values

The <form:option> tag

The **<form: option>** tag is used to represent an HTML **<option>** tag in any of the JSP pages of a Spring application. This tag is used when we need to add all the options to be **<form: select>** tag. Here, we need to add all the options inside the **<form: select>** tag. The following code snippet shows how to use the **<option>** tag in a JSP page:

```
<tr>
  <td>Department</td>
  <td><form:select path="department">
    <form:option value="technical" label="Technical" />
    <form:option value="non_technical" label="Non Technical" />
    <form:option value="r&d" label="R & D" />
  </form:select></td>
</tr>
```

The alternative code is below (with collection object)

```
<tr>
  <td>Department</td>
  <td><form:select path="department">
    <form:options items="${departmentMap}" />
  </form:select></td>
</tr>
```

The <form: textarea> tag

The **<form: textarea>** tag is used to represent an HTML **<textarea>** tag in any of the JSP pages of a Spring application. The following code snippet shows how to use the **<form: textarea>** tag in a JSP pages:

```
<td>Remarks :</td>
<td>
  <form:textarea path="remarks" rows="3"  cols="20"></form:textarea>
</td>
```

The <form: hidden> tag

The **<form: hidden>** tag is used to represent an HTML hidden field in a JSP page of a Spring application. The following code snippet shows how we use the **<hidden>** tag in JSP pages:

```
<form:hidden path="empId" />
```

The <form: errors> tag

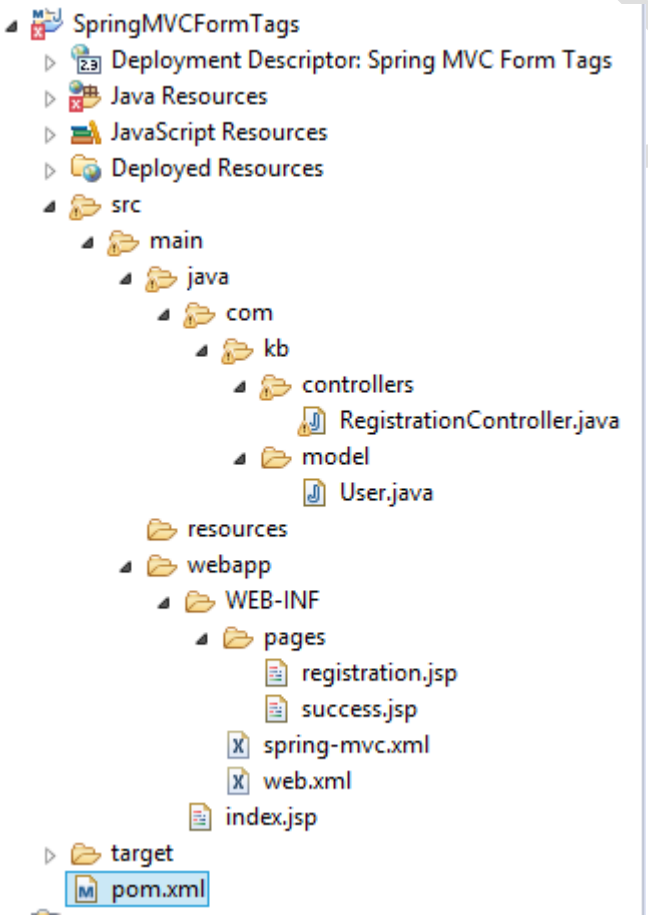
The **<form: errors>** tag is used to represent an HTML errors in a JSP of a
Spring application. This tag is used for accessing the error defined in an **org. springframework.validation.Validator** interface. For example, if we want to submit a form, and find all the validator error related to the name and password fields in that form, we have to define all the validation errors related to these fields in a **Validator** class, which must implements the **Validator** interface as follows:

```
<form:form method="POST" modelAttribute="user" action="register">
    <table>
      <tr>
        <td>Enter your name:</td>
        <td><form:input path="name" /></td>
        <td><form:errors path="name" cssStyle="color: #ff0000;"/></td>
      </tr>
      <tr>
        <td>Enter your mail:</td>
        <td><form:input path="email" /></td>
        <td><form:errors path="email" cssStyle="color: #ff0000;" /></td>
      </tr>
      <tr>
        <td>Select your gender</td>
        <td><form:radiobuttons path="gender" items="${genders}" /></td>
        <td><form:errors path="gender" cssStyle="color: #ff0000;"/></td>
      </tr>
    </table>
</form>
```

Let's see an example with all the above tags

We will create one Registration form and see how all the tags can be used

Step 1: Create one Maven Project like below



Step 2: Configure required dependencies is pom.xml like below

----------pom.xml----------

```
      <project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
      <modelVersion>4.0.0</modelVersion>
      <groupId>Spring</groupId>
      <artifactId>SpringMVCFormTags</artifactId>
      <packaging>war</packaging>
      <version>0.0.1-SNAPSHOT</version>
```

```
    <name>SpringMVCFormTags Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <properties>
        <org.springframework.version>4.2.0.RELEASE</org.springframework.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp.jstl</groupId>
            <artifactId>javax.servlet.jsp.jstl-api</artifactId>
            <version>1.2.1</version>
        </dependency>
        <dependency>
            <groupId>taglibs</groupId>
            <artifactId>standard</artifactId>
            <version>1.1.2</version>
        </dependency>
    </dependencies>
    <build>
        <finalName>SpringMVCFormTags</finalName>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.5.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Step 3: Configure DispatcherServlet in web.xml file like below

----------**web.xml**----------

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">

    <display-name>Spring MVC Form Tags</display-name>
```

```
	<!-- Spring MVC dispatcher servlet -->
	<servlet>
		<servlet-name>mvc-dispatcher</servlet-name>
		<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
		<init-param>
			<param-name>contextConfigLocation</param-name>
			<param-value>/WEB-INF/spring-mvc.xml</param-value>
		</init-param>
		<load-on-startup>1</load-on-startup>
	</servlet>
	<servlet-mapping>
		<servlet-name>mvc-dispatcher</servlet-name>
		<url-pattern>/</url-pattern>
	</servlet-mapping>

	<listener>
	<listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
	</listener>

	<!-- Loads Spring Security configuration file -->
	<context-param>
		<param-name>contextConfigLocation</param-name>
		<param-value>/WEB-INF/spring-mvc.xml</param-value>
	</context-param>
</web-app>
```

Step 4 : Create WebApplicationContext xml file like below

----------**spring-mvc.xml**----------

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
	xmlns:context="http://www.springframework.org/schema/context"
	xmlns:mvc="http://www.springframework.org/schema/mvc"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
	  http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
	  http://www.springframework.org/schema/context
	  http://www.springframework.org/schema/context/spring-context-3.2.xsd
	  http://www.springframework.org/schema/mvc
	  http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

	<context:component-scan base-package="com.kb.*" />
	<mvc:annotation-driven />

	<bean id="viewResolver"
		class="org.springframework.web.servlet.view.InternalResourceViewResolver">
		<property name="prefix" value="/WEB-INF/pages/" />
		<property name="suffix" value=".jsp" />
	</bean>
</beans>
```

Step 5 : Create User.java (model class) like below

----------User.java----------

```
package com.aits.model;

import java.util.List;

public class User {

	private String name;

	private String email;

	private String gender;
```

```
	private String password;

	private String passwordConfirm;

	private List<String> courses;

	private List<String> batches;

	private String hiddenMsg;

	//setters && getters
}
```

Step 6 : Create registration.jsp page to display registration form with input fields like below

----------**registration.jsp**----------

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<html>
<head>
<title>Spring MVC form tags</title>
</head>
<body>
	<h2>Fill below form to register</h2>
	<form:form method="POST" modelAttribute="user" action="register">
		<table>
			<tr>
				<td>Enter your name:</td>
				<td><form:input path="name" /></td>
				<td><form:errors path="name" cssStyle="color: #ff0000;"/></td>
			</tr>
			<tr>
				<td>Enter your mail:</td>
				<td><form:input path="email" /></td>
				<td><form:errors path="email" cssStyle="color: #ff0000;"/></td>
			</tr>
			<tr>
				<td>Enter your gender</td>
				<td><form:radiobuttons path="gender" items="${genders}"/></td>
				<td><form:errors path="gender" cssStyle="color:
#ff0000;"/></td>
			</tr>
			<tr>
				<td>Enter a password:</td>
				<td><form:password path="password" showPassword="true"/></td>
				<td><form:errors path="password" cssStyle="color:
#ff0000;"/></td>
			<tr>
				<td>Confirm your password:</td>
				<td><form:password path="passwordConfirm"
showPassword="true"/></td>
				<td><form:errors path="passwordConfirm"
					cssStyle="color: #ff0000;"/></td>
			</tr>
			<tr>
				<td>Choose your Batches:</td>
				<td><form:checkboxes path="batches" items="${batches}"/></td>
				<td><form:errors path="batches" cssStyle="color:
#ff0000;"/></td>
			</tr>
			<tr>
				<td>Please select your courses:</td>
				<td><form:select path="courses">
					<form:option value="" label="Please Select"/>
					<form:options items="${courses}"/>
				</form:select></td>
```

```
                        <td><form:errors path="courses" cssStyle="color:
#ff0000;"/></td>
                  </tr>
                  <tr>
                        <form:hidden path="hiddenMsg"/>
                  </tr>
                  <tr>
                        <td><input type="submit" name="submit" value="Register"></td>
                  </tr>
                  <tr>
            </table>
      </form:form>
</body>
</html>
```

- path is the spring form tag attribute used to bind the form field with the model class.
- <form: errors> tag is used to specify error to be displayed for the corresponding field.
- We can specify the list of strings directly using items as we used in < form: checkboxes> and < form:select>
- List specified with items is used to display multiple values and path is used to bind the selected value into the model class variable.

Step 7 : Create RegistrationController.java file in src/main/java folder to process user requests

----------RegistrationController.java----------

```
package com.aits.controllers;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.aits.model.User;

/**
 *
 * @author Ashok Bollepalli
 * This controller is used to Handle User registration
 * screen related actions
 *
 */

@Controller
public class RegistrationController {

      @RequestMapping(value = "/displayRegistrationPage", method = RequestMethod.GET)
      public String displayUserPage(Model model) {
            User user = new User();
            user.setHiddenMsg("Ashok IT School");
            model.addAttribute("user", user);
            initializeFormValues(model);
            return "registration";
      }

      @RequestMapping(value = "/register", method = RequestMethod.POST)
      public String displayUserDetails(@ModelAttribute User user, Model model) {
            model.addAttribute("user", user);
            return "success";
      }

      private void initializeFormValues(Model model) {
            List<String> courses = new ArrayList<String>();
```

```
            courses.add("Java");
            courses.add("J2EE");
            courses.add("Spring");
            courses.add("Hibernate");
            courses.add("Jquery");
            model.addAttribute("courses", courses);

            List<String> genders = new ArrayList<String>();
            genders.add("Male");
            genders.add("Female");
            model.addAttribute("genders", genders);

            List<String> batches = new ArrayList<String>();
            batches.add("morning");
            batches.add("evening");
            model.addAttribute("batches", batches);

      }
}
```

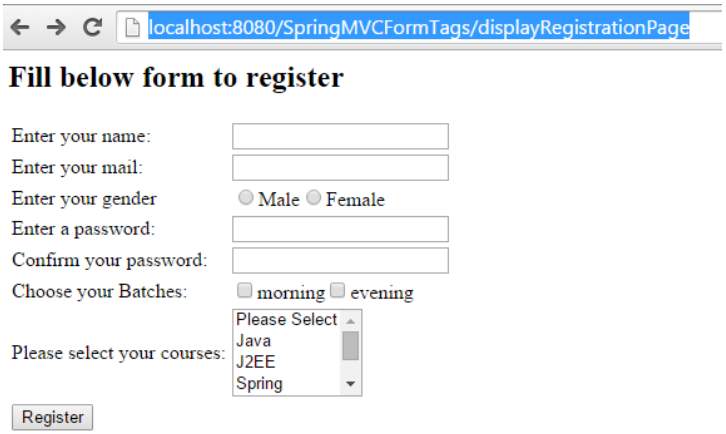
In the controller we have added method to display the Registration page and success page on submit. We have also written a method to initialize the form with all the required values.

Step 8 : Create success.jsp page to display captured form data as a response

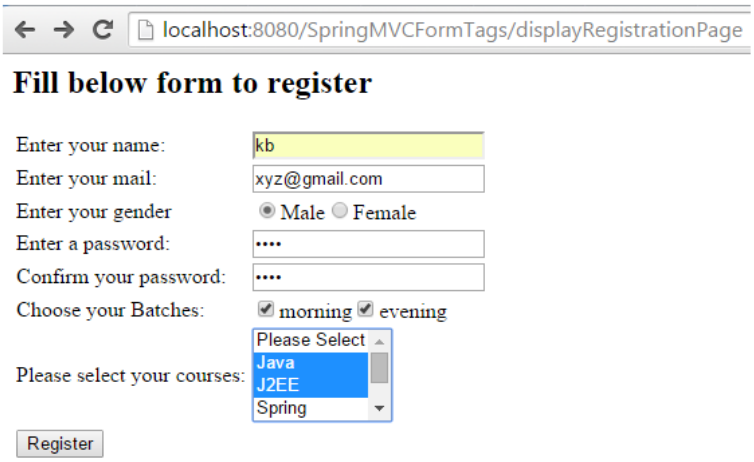
----------**success.jsp**----------

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
      Hey ${user.name} , you are successfully registered.
      <br> You have chosen the below courses:
      <br>
      <c:forEach var="course" items="${user.courses}">
            <c:out value="${course}" />
            <br>
      </c:forEach>
      <br> You have chosen the below batches:
      <br>
      <c:forEach var="batch" items="${user.batches}">
            <c:out value="${batch}" />
            <br>
      </c:forEach>
      <br> Your hidden name is ${user.hiddenMsg}
</body>
</html>
```

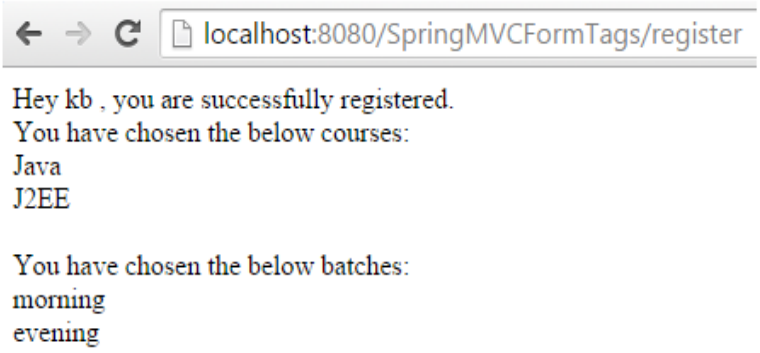
Step 9: Deploy the application into server and access using below URL



Step 10: Fill the form



Step 11: Click on Register button, below screen will be displayed



===000===